CS 161 Lab Worksheet: Week 2 (2 pts)

Design and Testing (Assignment #2)
Design is very important when developing programs and there are a variety of ways to approach it. You may draw pictures, write it out in prose or structured text, use pseudo code, and more! The point of design is to give you a blueprint to follow while you are coding. This saves time debugging your program as you can catch mistakes early. It is better to spend one hour of designing than it is to spend five hours debugging.

George Polya developed a well-known model for problem solving in mathematics that is based on these 4 principles.

- Understanding the problem. (*Recognizing what is asked*.)
- Devising a plan. (*Responding to what is asked*.)
- Carrying out the plan. (*Developing the result of the response*.)
- Looking back. (*Checking. What does the result tell me? Did I do it right?*)

Polya's steps 1, 2, and 4 do not directly deal with writing the solution (in programming that is the C++ code itself), but rather, the steps you need to make sure you write a correct solution/program that solves the given problem statement. With this said, make yourself familiar assignment #2 and Polya's steps 1, 2, and 4.

Understanding the Problem
In your own words, explain what YOU think the problem is asking you to do. Document your uncertainties about the problem and anything else that you feel was unclear or vague. This is to ensure that YOUR understanding matches MY understanding of the problem.☺ State any assumptions you are making and all the user and functional requirements.

Devising a Plan/Design
Provide an algorithm/pseudo code to help solve the problem. In addition, draw pictures/flow charts to help you devise your plan, as well as any other design decisions you make, such as how to manage your time, how to decompose the problem, where to start first, etc.

Looking back/Testing
This includes any checking/self-reflection you did while solving the problem, which includes using a calculator to make sure the output is correct, testing to make sure your code executes correctly and behaves the way you expect under specific circumstances, using sources of information to make sense of the results, etc. However, you need to think about the input prior to implementation!!!

Please see an example of this document: Polya_template.pdf

Using Assignment #2, get into pairs to answer the following questions on a piece of paper with your names written on the paper to give your TA:

(1 pt) Understanding the Problem – Do you understand everything in the problem? Do you understand what is meant by "there must be **at least two paths with a depth of 3 nested loops** to complete the adventure" and "there must be **an element of chance** that would change a user's chosen path"?

What assumptions are you making about the problem you are solving, user input, etc.

What are the user requirements? What are the functional requirements?

(.5 pt) Design – What are the steps to create this adventure game? What are the steps for handling bad input? Begin developing a flow-chart and/or pseudo-code for the adventure each person in the group wants to create.

(.5 pt) Testing – Create a test plan with the test cases (**bad, good, and edge cases**). What do you hope to be the expected results?